Please replace the paragraph beginning at page 15, line number 18, with the following rewritten paragraph:

Turning now to Fig. 13, the maintenance step 490 of Fig. 8B is shown in detail. The maintenance step 490 is an event driven process and thus receives a software trigger event (step 492). Based on the trigger event, the process of Fig. 13 determines various possible events, including a check for update event 494, a protect software event (step 496), a software recovery (step 498) event, a check removal event (step 500), and an examine system event (step 502). From steps 494-502, the triggering event is reported (step 504) before the process of Fig. 13 exits (step 506).

A13

In the claims:

Cancel claim 1.

Add claims 2 through 22:

Sub B4

2. (New) A computer-implemented vault for archiving software components, where only a single instance of each component that is multiply-used is stored in the vault, comprising:

unique instances of the one or more software components;

A14 Cont

an access controller for performing a direct, random access retrieval of the one or more software components from the vault; and

a post controller for performing a direct, random access insertion of a software component to the vault wherein the post controller generates a unique key from the new component and optimizes storage if the unique key exists.

3. (New) A computer-implemented vault for archiving software components, where only a single instance of each component that is multiply-used is stored in the vault, comprising:

unique instances of the one or more software components;

an access controller for performing a direct, random access retrieval of the one or more software components from the vault; and

a client coupled to the vault, the client having a physical software component residing on the client, the client generating a key from the physical software component.

4. (New) A computer-implemented vault for archiving software components, where only a single instance of each component that is multiply-used is stored in the vault, comprising:

unique instances of the one or more software components;

an access controller for performing a direct, random access retrieval of the one or more software components from the vault;

one or more secondary vaults coupled to the vault; and

a fault-tolerant rollover system for sequentially searching each vault for the presence of a target software component.

B4

A14 Cont

5. (New) The computer-implemented vault of claim 4, wherein the secondary vaults are ordered based on accessibility of the vaults.

6. (New) The computer-implemented vault of claim 4, further comprising a client for generating a key, the client applying the key to recover the target software component from the most accessible of the vaults.

7. (New) The computer-implemented vault of claim 6, wherein the client uses a metadata description to generate the key.

8. (New) The computer-implemented vault of claim 6, wherein the search of a determined vault fails to locate the target software component, and wherein the client skips the determined vault and modifies the search order of the vaults in recovering the target software component.

9. (New) A computer-implemented vault for archiving software components, where only a single instance of each component that is multiply-used is stored in the vault, comprising:

means for storing unique instances of the one or more software components on the vault;

access means for performing a direct, random access retrieval of the one or more software components from the vault; and

a post means for performing a direct, random access insertion of a software component to the vault wherein the post means generates a unique key from the new component and optimizes storage if the unique key exists.

10. (New) A computer-implemented vault for archiving software components, where only a single instance of each component that is multiply-used is stored in the vault, comprising:

means for storing unique instances of the one or more software components on the vault;

access means for performing a direct, random access retrieval of the one or more software components from the vault; and

a client coupled to the vault, the client having a physical software component residing on the client, the client generating a key from the physical software component.

11. (New) A computer-implemented vault for archiving software components, where only a single instance of each component that is multiply-used is stored in the vault, comprising:

means for storing unique instances of the one or more software components on the vault;

access means for performing a direct, random access retrieval of the one or more software components from the vault;

one or more secondary vaults coupled to the vault; and

means for sequentially searching each vault for the presence of a target software component.

12. (New) The computer-implemented vault of claim 10, wherein the secondary vaults are ordered based on accessibility of the vaults.

13. (New) The computer-implemented vault of claim 10, further comprising a client for generating a key, the client having a means for applying the key to recover the target software component from the most accessible of the vaults.

14. (New) The computer-implemented vault of claim 13, wherein the client uses a metadata description to generate the key.

15. (New) The computer-implemented vault of claim 13, wherein the search of a determined vault fails to locate the target software component, and wherein the client skips the determined vault and modifies the search order of the vaults in recovering the target software component.

16. (New) A method for archiving software components where only a single instance of each component that is multiply-used is stored in a vault, comprising the steps of:

storing unique instances of the one or more software components in the vault; and

performing a direct, random access retrieval of the one or more software components from the vault; and

performing a direct, random access insertion of a software component to the vault wherein said step of performing an insertion generates a unique key from the new component and optimizes storage if the key exists.

17. (New) A method for archiving software components where only a single instance of each component that is multiply-used is stored in a vault, comprising the steps of:

storing unique instances of the one or more software components in the vault;

performing a direct, random access retrieval of the one or more software components from the vault; and

generating a key from a physical software component residing on a client coupled to the vault.

18. (New) A method for archiving software components where only a single instance of each component that is multiply-used is stored in a vault, and wherein one or more secondary vaults are coupled to the vault, comprising the steps of:

storing unique instances of the one or more software components in the vault; and

performing a direct, random access retrieval of the one or more software components from the vault; and

sequentially searching each vault for the presence of a target software component.

19. (New) The method of claim 18, wherein the secondary vaults are ordered based on accessibility of the vaults.

20. (New) The method of claim 17, further comprising a client for generating a key, the client applying the key to recover the target software component from the most accessible of the vaults.

21. (New) The method of claim 20, wherein the client uses a metadata description to generate the key.

22. (New) The method of claim 20, wherein the search of a determined vault fails to locate the target software component, and wherein the client skips the determined vault and modifies the search order of the vaults in recovering the target software component.